Apple IIGS
#3: Window Information Bar Use

Revised by: Dave Lyons January 1991
Written by: Dan Oliver October 1986

This Technical Note details the use of a window's information bar, including a code sample which places a menu in an information bar.

Changes since November 1988: Added a note about the current Resource Application when inside an `InfoDefProc` procedure, and information about information bars and `NewWindow2`.

Apple IIGS window information bars are not as straightforward as other window features, and one reason for this is the small amount of space originally allocated for their processing. If you feel your application can benefit from the use of information bars, you can implement them, and this Technical Note explains how to do it and includes some suggestions for their use. The code samples below demonstrate how to place a menu bar in an information bar, but your use of information bars is not limited to those described here.

Information Bar Initialization

You can create an information bar in a window when you create the window by setting the following fields in the parameter list you pass to `NewWindow`:

`wFrame` Set bit 4.

`wInfoHeight` Set to the height of the information bar (should not exceed window height).

`wInfoDefProc` Set to the address of the information bar definition procedure (see below).

If you create a window as visible, the Window Manager will call your information bar definition procedure (`InfoDefProc`) before returning from `NewWindow`. If you have to create the contents of the information bar after the window, you will have a problem since the Window Manager will expect your `InfoDefProc` to draw things which do not yet exist. You can solve this problem by creating the window as invisible, creating the contents of the information bar, then showing the window. Another solution would be to detect, in the `InfoDefProc`, that the contents of the information bar do not yet exist.

NewWindow2, however, does not let you override the information bar drawing procedure in the template. If you pass a window template in a resource, creating the window as visible crashes (since the address of your information bar drawing procedure cannot possibly be in the window template resource). Instead, create the window as invisible and call SetInfoDraw to set the address of the information bar drawing procedure **before** calling ShowWindow.

Below is an example of initializing a window's information bar to contain a menu bar. The three key fields of the parameter list which you pass to NewWindow are as follows:

wFrame — Set bit 4 = 1 and bit 5 = 0 for an invisible window; the other bits do not affect the information bar, so you can set them as you wish.

wInfoHeight — Assuming you are using a system menu bar and initializing it before the window, set to the height FixMenuBar returned when you created the system menu bar. If you would rather use an absolute value, which we do not advise, you could use 14 which should be about right for the current system font.

wInfoDefProc — Set to the address of the InfoDefProc, in this case draw_info.

After you create the window, but before you show it, you can create the menu bar to place in the information bar. The code to create the menu bar might look like the following:

```
window              Direct page location that contains pointer to window's port.
;
; --- Create a menu bar ----------------------------------------------------
------
;
             pha                        Space for result.
             pha
             pea    $FFFF               Set "use current port" flag.
             pea    $FFFF
             _NewMenuBar                Create a menu bar.
             pla                        Get returned menu bar handle.
             sta    <menuBar            Remember menu bar handle.
             pla
             sta    <menuBar+2
;
;
; --- Store menu bar's handle in the window's InfoRefCon -------------------------
------
;
             pei    <menuBar+2          Pass menu bar handle.
             pei    <menuBar
             pei    <window+2           Window to set refCon.
             pei    <window
             _SetInfoRefCon             Store menu bar handle in window's infoRefCon.
;
;
; --- Make the window's menu bar the current menu bar ----------------------------
------
;
             pei    <menuBar+2          Pass menu bar handle.
             pei    <menuBar
             _SetMenuBar                Make new menu bar the current menu bar.

;
```

```
;
; --- Get the RECT of the window's information bar ----------------------------------
------
;
             pea    tempRect|-16        Pass pointer of RECT.
             pea    tempRect
             pei    <window+2           Pass pointer of window.
             pei    <window
             _GetRectInfo               tempRect = interior RECT of window's Info Bar.


; --- Dereference menu bar handle ---------------------------------------------------
------
;
             ldy    #2
             lda    [menuBar],y
             tay
             lda    [menuBar]
             sta    <menuBar            Now menuBar is the pointer to the Menu Bar.
             sty    <menuBar+2
;
;
; --- Set size of menu bar ----------------------------------------------------------
------
;
;
             lda    <tempRect+y1
             dec    a                   Overlap top side.
             ldy    #CtlRect+y1
             sta    [menuBar],y
;
             lda    <tempRect+x1
             dec    a                   Overlap left side.
             ldy    #CtlRect+x1
             sta    [menuBar],y
;
             lda    <rect+y2
             inc    a                   Overlap bottom side.
             ldy    #CtlRect+y2
             sta    [menuBar],y
;
;
; --- Set flag to tell Menu Manager to draw menu in current port --------------------
------
;
             ldy    #CtlOwner+2         Set high bit in CtlOwner.
             lda    [menuBar],y
             ora    #$8000
             sta    [menuBar],y
;
;
; --- Create the menus and add them to the window's menu bar ------------------------
------
;
             lda    #4
loop         pha                        Save index into menu list.
             tay                        Switch index to Y.
;
             pha                        Space for return value.
             pha
             lda    menu_list+2,y       Pass address of menu/item lines.
             pha
             lda    menu_list,y
             pha
             _NewMenu
;                                       Menu handle already on stack.
```

```
            pea    0                    Insert menu list at front of list.
            _InsertMenu                 Add my menus to the system menu bar.
;
            pla
            sec
            sbc    #4
            bpl    loop
;
;
; --- Initialize the size of the menu bar and menus ---------------------------------
------
;
            pha                         Space for returned bar height.
            _FixMenuBar                 Fix up positions in the menu bar.
            pla                         Discard height of menu bar.
;
;
; --- Restore the system menu bar as the current menu -------------------------------
------
;
            pea    0                    Pass flag for system menu bar.
            pea    0
            _SetMenuBar                 Make system menu bar current.
```

The window's menu bar is now initialized, and you can make the window visible with a call to `ShowWindow`; the `InfoDefProc` will draw the menu bar.

Information Bar Definition Procedure (InfoDefProc)

The `InfoDefProc` is slightly misleading; it is only responsible for drawing the interior, above the background, of the information bar. The `InfoDefProc` is not responsible for defining the information bar, drawing the frame and background, testing for hits, or tracking the user. The `InfoDefProc` is located inside your application, and the Window Manager calls it whenever it needs to draw the part of the window frame that contains the information bar. Each window with an information bar can have its own `InfoDefProc`, or they can all share a common `InfoDefProc`. When the Window Manager calls your `InfoDefProc`, it sets the proper port, the Window Manager's port, and the proper state, an origin local to the window frame and clipped to any windows above it. The direct page and data bank are not defined and should be considered unknown.

The Window Manager passes your `InfoDefProc` the following information:

- Pointer to the information bar's interior rectangle (less frame), local coordinates.
- Value of the window's `wInfoRefCon`, set and used only by your application.
- Pointer to the window's port (do **not** switch to this port for drawing).

Note: When the Window Manager calls your `InfoDefProc`, there is no guarantee that the current Resource Application is set to the value you expect. If your `InfoDefProc` makes Resource Manager calls, directly or indirectly, be sure to save, set, and restore the Resource Application using `GetCurResourceApp` and `SetCurResourceApp`.

A window that has an information bar containing a menu bar (handle stored in the window's `InfoRefCon`) might have a `InfoDefProc` as follows:

```
draw_info     START
;
theWindow     equ    6                  Offset to the information bar owner window.
infoRefCon    equ    theWindow+4        Offset to the window's information bar RefCon.
infoRect      equ    infoRefCon+4       Offset to the information bar's enclosing
RECT.
;
              phd                       Save original direct page.
              tsc                       Switch to direct page in stack.
              tcd
;
;
; --- Draw the window's menu bar in the window's information bar ----------------
------
;
              pei    infoRefCon+2       Pass handle of window's menu bar handle.
              pei    infoRefCon
              _SetMenuBar               Make the window's menu bar the current menu
bar.
;
              _DrawMenuBar              Draw the window's menu bar, as requested.
;
              lda    #0                 Zero is the flag for the system menu bar.
              pha
              pha
              _SetMenuBar               Make the system menu bar current again.
```

```
;
;
; --- Remove input parameters from the stack ----------------------------------------
------
;            ldx    #12
             ply                         Pull original direct page off stack, save in
Y.
;
             tsc                         Move direct page point to stack.
             tcd
             lda    2,s                  Move   return   address   down   over   input
parameters.
             sta    2,x
             lda    0,s
             sta    0,x
;
             tsc                         Adjust stack for stripped input parameters.
             phx                         Number of bytes of input parameters.
             clc
             adc    1,s                  Add  number  of  input  parameters  to  stack
pointer.
             tcs                         And reset stack.
;
             tya                         Restore original direct page.
             tcd
;
             rtl                         Return to Window Manager.
             END
```

Information Bar Environment

An information bar is part of a window's frame, that is, not part of the window's content region. Because it is part of the frame, an information bar is in the Window Manager's port, so before an interaction (drawing or mouse selecting), the proper port (Window Manager's) must be in the proper state. The proper state means the origin must be at the window's upper-left corner and clipped to any windows above.

When the Window Manager calls the `InfoDefProc` it sets the proper port to the proper state; however, to interact with the information bar outside the `InfoDefProc`, you must set the proper port to the proper state. You can accomplish this with a call to `StartInfoDrawing`. When the interaction is completed, you must allow the Window Manager to return its port to a general state via a call to `EndInfoDrawing`. You are in a special state that requires some constraints (discussed later) between the calls to `StartInfoDrawing` and `EndInfoDrawing`.

Here is an example of interacting with our window's menu bar.

```
;
poll         pha                         Space for return value.
             pea    %0000111101101110    Pass event mask to use.
             pea    TaskRec|-16          Pass pointer to Task record.
             pea    TaskRec
             _TaskMaster
             pla                         Get returned value.
             beq    poll                 Does event need further processing?
;
```

```
;
; --- Handle button down in window's information bar --------------------------------
------
;
            cmp    #InInfo             In Information bar?
            bne    poll
;
            pha                        Space for result.
            pha
            lda    TaskRec+TaskData+2  Pass pointer of window.
            pha
            lda    TaskRec+TaskData
            pha
            _GetInfoRefCon             Get menu bar handle from window's InfoRefCon.
            pla
            sta    menuBar
            pla
            sta    menuBar+2
;
;
; --- Switch to proper port in proper coordinate system -----------------------------
------
;
            pea    tempRect|-16        Pass pointer to RECT to store info bar RECT.
            pea    tempRect
            lda    TaskRec+TaskData+2  Pass pointer of window.
            pha
            lda    TaskRec+TaskData
            pha
            _StartInfoDrawing
;
;
; --- Handle menu selection from window's menu bar ----------------------------------
------
;
            pea    TaskRec|-16         Pass pointer to Task record for MenuSelect.
            pea    TaskRec
            pei    menuBar+2           Pass handle of menu bar.
            pei    menuBar
            _MenuSelect                Let user make selection.
;
            lda    event+TaskData      Get the item's ID number.
            beq    exit                Was a selection made?
;
            _EndInfoDrawing            Switch back to original port.

;
;         (Handle the menu selection.)
;
;            The EndInfoDrawing followed by the StartInfoDrawing call is only
;            needed when code between them calls the Window Manager.
;
            pea    tempRect|-16        Pass pointer to RECT to store info bar RECT.
            pea    tempRect
            lda    TaskRec+TaskData+2  Pass pointer of window.
            pha
            lda    TaskRec+TaskData
            pha
            _StartInfoDrawing          Switch to the proper port in the proper state.
;
            pea    0                   Pass unhilite flag.
            lda    TaskRec+TaskData+2  Pass menu's ID number.
            pha
            _HiliteMenu                Unhilite menu's title.
;
```

```
;
; --- Clean up and return to polling ------------------------------------------
------
;
exit          _EndInfoDrawing           Switch back to original port.
;
              pea    0                  Make system menu bar current.
              pea    0
              _SetMenuBar
;
              jmp    poll               Return to polling user.
;
```

Information Bar Shutdown

When the Window Manager closes the window, it is up to you to resolve any shutdown necessities associated with the information bar. Using our window menu bar example, the close window might look like the following:

```
;
              pei    menuBar+2          Pass handle of menu bar
              pei    menuBar
              _SetMenuBar
;
              pha                       Space for returned menu handle.
              pha
              pea    2                  ID number of second menu.
              _GetMHandle               Get the menu's handle.
              _DisposeMenu              Free menu record and associated data.
;
              pha                       Space for returned menu handle.
              pha
              pea    1                  ID number of first menu.
              _GetMHandle               Get the menu's handle.
              _DisposeMenu              Free menu record and associated data.
;
              pea    0                  Make system menu bar current.
              pea    0
              _SetMenuBar
;
              pha                       Space for menu bar's handle.
              pha
              pei    <window+2          Pass pointer of window to close.
              pei    <window
              _GetInfoRefCon            Get the InfoRefCon from the window.
              _DisposeHandle            Free menu bar record.
;
              pei    <window+2          Pass pointer of window to close.
              pei    <window
              _CloseWindow              Now the window can be closed.
;
```

The type of shutdown you use depends upon the contents of the information bar.

Why didn't I put a `DisposeMenuBar` call in the Menu Manager? I didn't think of it until a week too late. Sorry.

Other Information Bar Uses

The following suggestions are only theories and have not been tested.

- Display text information, as in Finder windows.
- Split window. Like the content region, the information bar could be large enough to hold data.
- Hold controls. You could scroll data in the content region while keeping the controls which affect the display in place and within the user's reach. (Note: The Control Manager does not know about information bars. If you want to draw and track objects in information bars, you have to do it yourself using QuickDraw II calls.)

Further Reference

- *Apple IIGS Toolbox Reference*, Volumes 1-3
- Apple IIGS Technical Note #83, Resource Manager Stuff